

CLAIMS

What is claimed is:

- 1 1. A method comprising:
 - 2 re-compiling a function when a field watch for a field is activated, the
 - 3 function including a byte code sequence having a field byte code that accesses or
 - 4 modifies the field, the recompiled function providing a native code and occupying
 - 5 a code space;
 - 6 generating an instrumentation code corresponding to the field watch of the
 - 7 field; and
 - 8 inserting the instrumentation code to the native code.
- 1 2. The method of claim 1 further comprising:
 - 2 guarding execution of the instrumentation code if the field watch is not
 - 3 activated.
- 1 3. The method of claim 1 wherein generating the instrumentation
- 2 code comprises:
 - 3 executing a field watch sequence if the field watch is activated.
- 1 4. The method of claim 1 wherein executing the field watch sequence
- 2 comprises:

3 saving live global state, the live global state corresponding to an active
4 register;
5 executing an event hook function for an event corresponding to the field
6 watch; and
7 restoring the live global state.

1 5. The method of claim 4 wherein saving the live global state
2 comprises:

3 pushing the live global state onto a stack.

1 6. The method of claim 4 wherein executing the event hook function
2 comprises:

3 passing an argument corresponding to the field; and

4 calling a run-time library function related to the event.

1 7. The method of claim 5 wherein restoring the live global state
2 comprises:

3 retrieving the live global state from the stack.

1 8. The method of claim 1 wherein inserting the instrumentation code
2 comprises:

3 inserting the instrumentation code in a stub at end of the code space.

1 9. The method of claim 2 wherein guarding execution of the
2 instrumentation code comprises:
3 updating an offset of a jump instruction to the stub when the field watch is
4 activated.

1 10. The method of claim 1 wherein guarding execution of the
2 instrumentation code comprises:
3 replacing a no-op sequence with a jump instruction to the stub.

1 11. The method of claim 9 further comprising:
2 clearing the field watch by replacing the offset with a zero offset.

1 12. The method of claim 10 further comprising:
2 clearing the field watch by replacing the jump instruction with the no-op
3 sequence.

1 13. The method of claim 1 wherein the function is a Java method.

1 14. The method of claim 1 wherein the field is a Java field in a Java
2 virtual machine.

1 15. The method of claim 4 wherein the event hook function is
2 compatible with a Java Virtual Machine Debug Interface (JVMDI).

1 16. A computer program product comprising:
2 a machine useable medium having computer program code embedded
3 therein, the computer program product having:
4 computer readable program code to re-compile a function when a
5 field watch for a field is activated, the function including a byte code
6 sequence having a field byte code that accesses or modifies the field, the
7 recompiled function providing a native code and occupying a code space,
8 computer readable program code to generate an instrumentation
9 code corresponding to the field watch of the field, and
10 computer readable program code to insert the instrumentation code
11 to the native code.

1 17. The computer program product of claim 16 further comprising:
2 computer readable program code to guard execution of the instrumentation
3 code if the field watch is not activated.

1 18. The computer program product of claim 16 wherein the computer
2 readable program code to generate the instrumentation code comprises:
3 computer readable program code to execute a field watch sequence if the
4 field watch is activated.

1 19. The computer program product of claim 16 wherein the computer
2 readable program code to execute a field watch sequence comprises:

3 computer readable program code to save live global state, the live global
4 state corresponding to an active register;

5 computer readable program code to execute an event hook function for an
6 event corresponding to the field watch; and

7 computer readable program code to restore the live global state.

1 20. The computer program product of claim 19 wherein the computer
2 readable program code to save the live global state comprises:

3 computer readable program code to push the live global state onto a stack.

1 21. The computer program product of claim 19 wherein the computer
2 readable program code to execute the event hook function comprises:

3 computer readable program code to pass an argument corresponding to the
4 field; and

5 computer readable program code to call a run-time library function related
6 to the event.

1 22. The computer program product of claim 20 wherein the computer
2 readable program code to restore the live global state comprises:

3 computer readable program code to retrieve the live global state from the
4 stack.

1 23. The computer program product of claim 16 wherein the computer
2 readable program code to insert the instrumentation code comprises:

3 computer readable program code to insert the instrumentation code in a
4 stub at end of the code space.

1 24. The computer program product of claim 16 wherein the computer
2 readable program code to guard execution of the instrumentation code comprises:

3 computer readable program code to update an offset of a jump instruction
4 to the stub when the field watch is activated.

1 25. The computer program product of claim 16 wherein the computer
2 readable program code to guard execution of the instrumentation code comprises:

3 computer readable program code to replace a no-op sequence with a jump
4 instruction to the stub.

1 26. The computer program product of claim 24 further comprising:

2 computer readable program code to clear the field watch by replacing the
3 offset with a zero offset.

1 27. The computer program product of claim 25 further comprising:

2 computer readable program code to clear the field watch by replacing the
3 jump instruction with the no-op sequence.

1 28. The computer program product of claim 16 wherein the function is
2 a Java method.

1 29. The computer program product of claim 16 wherein the field is a
2 Java field in a Java virtual machine.

1 30. The computer program product of claim 19 wherein the event hook
2 function is compatible with a Java Virtual Machine Debug Interface (JVMDI).

1 31. A system comprising:
2 a processor;
3 a memory coupled to the processor to store instruction code, the
4 instruction code, when executed by the processor, causing the processor to:
5 re-compile a function when a field watch for a field is activated,
6 the function including a byte code sequence having a field byte
7 code that accesses or modifies the field, the re-compiled function
8 providing a native code and occupying a code space,
9 generate an instrumentation code corresponding to the field watch
10 of the field, and
11 insert the instrumentation code to the native code.

1 32. The system of claim 31 the instruction code further causing the
2 processor to:

3 guard execution of the instrumentation code if the field watch is not
4 activated.

1 33. The system of claim 31 wherein the instruction code causing the
2 processor to generate the instrumentation code causes the processor to:
3 execute a field watch sequence if the field watch is activated.

1 34. The system of claim 31 wherein the instruction code causing the
2 processor to execute a field watch sequence causes the processor to:
3 save live global state, the live global state corresponding to an active
4 register;
5 execute an event hook function for an event corresponding to the field
6 watch; and
7 restore the live global state.

1 35. The system of claim 32 wherein the instruction code causing the
2 processor to guard execution of the instrumentation code causes the processor to:
3 update an offset of a jump instruction to the stub when the field watch is
4 activated.

1 36. The system of claim 32 wherein the instruction code causing the
2 processor to guard execution of the instrumentation code causes the processor to:
3 replace a no-op sequence with a jump instruction to the stub.

1 37. The system of claim 31 wherein the function is a Java method.

1 38. The system of claim 31 wherein the field is a Java field in a Java
2 virtual machine.

1 39. The system of claim 34 wherein the event hook function is
2 compatible with a Java Virtual Machine Debug Interface (JVMDI).